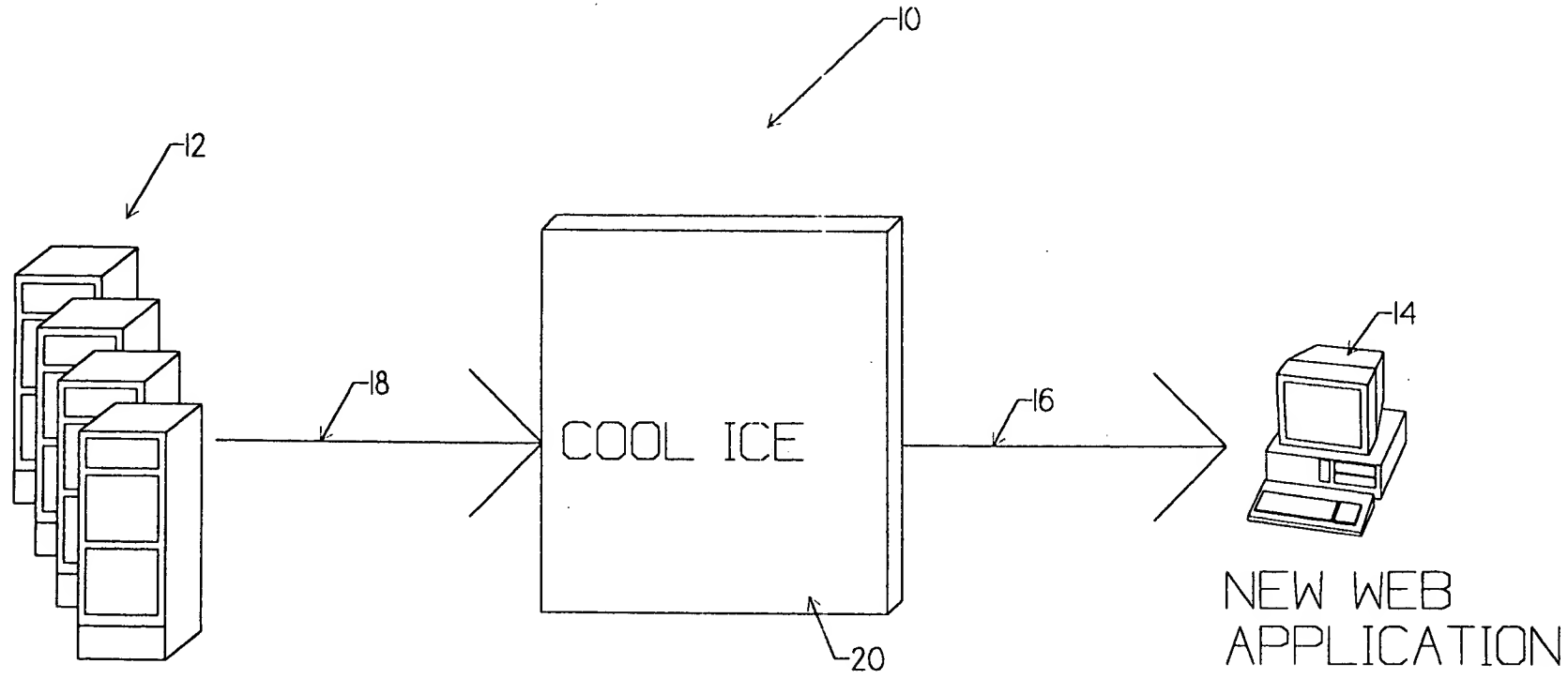

FIG. 1

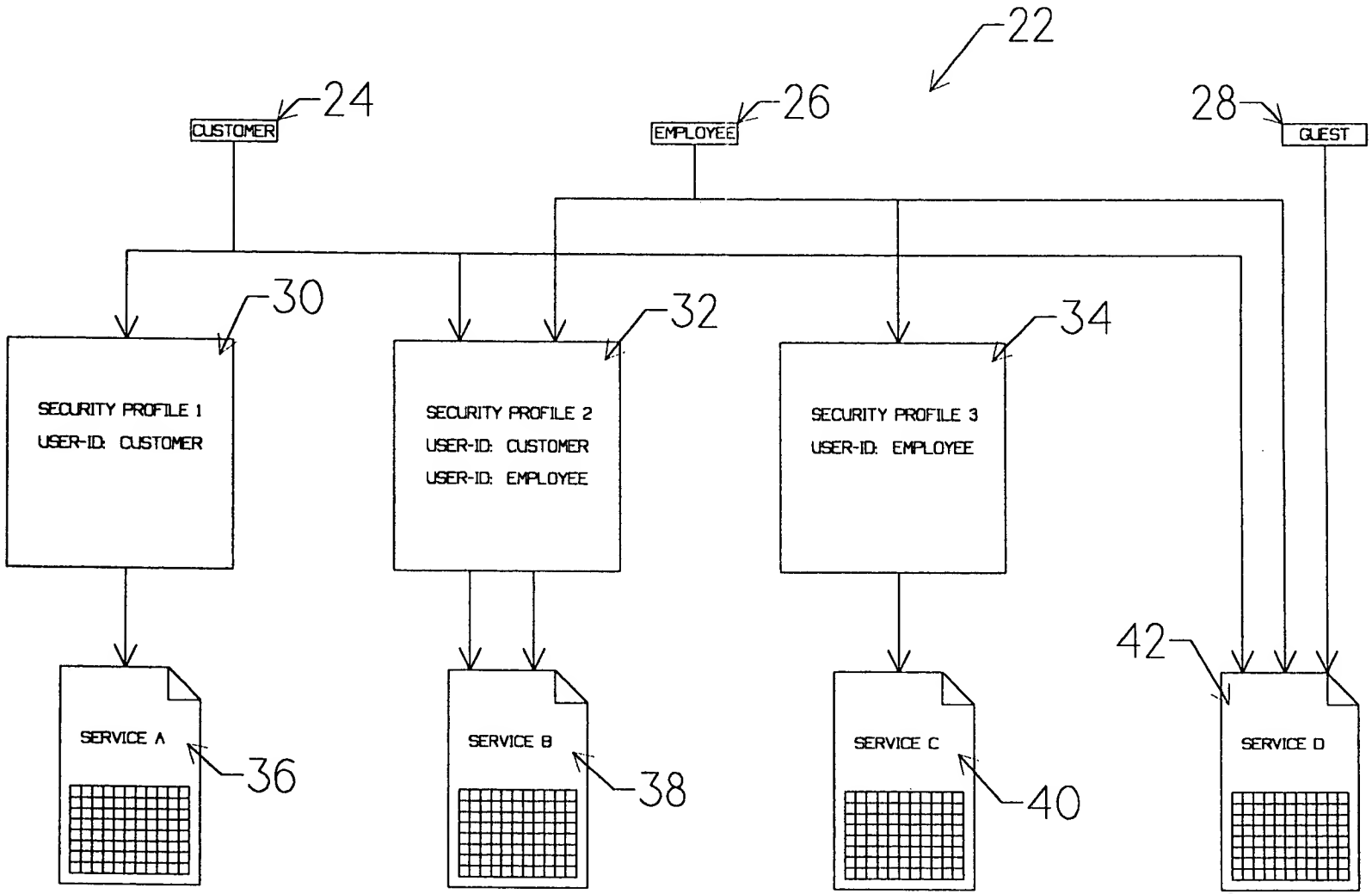

FIG. 2

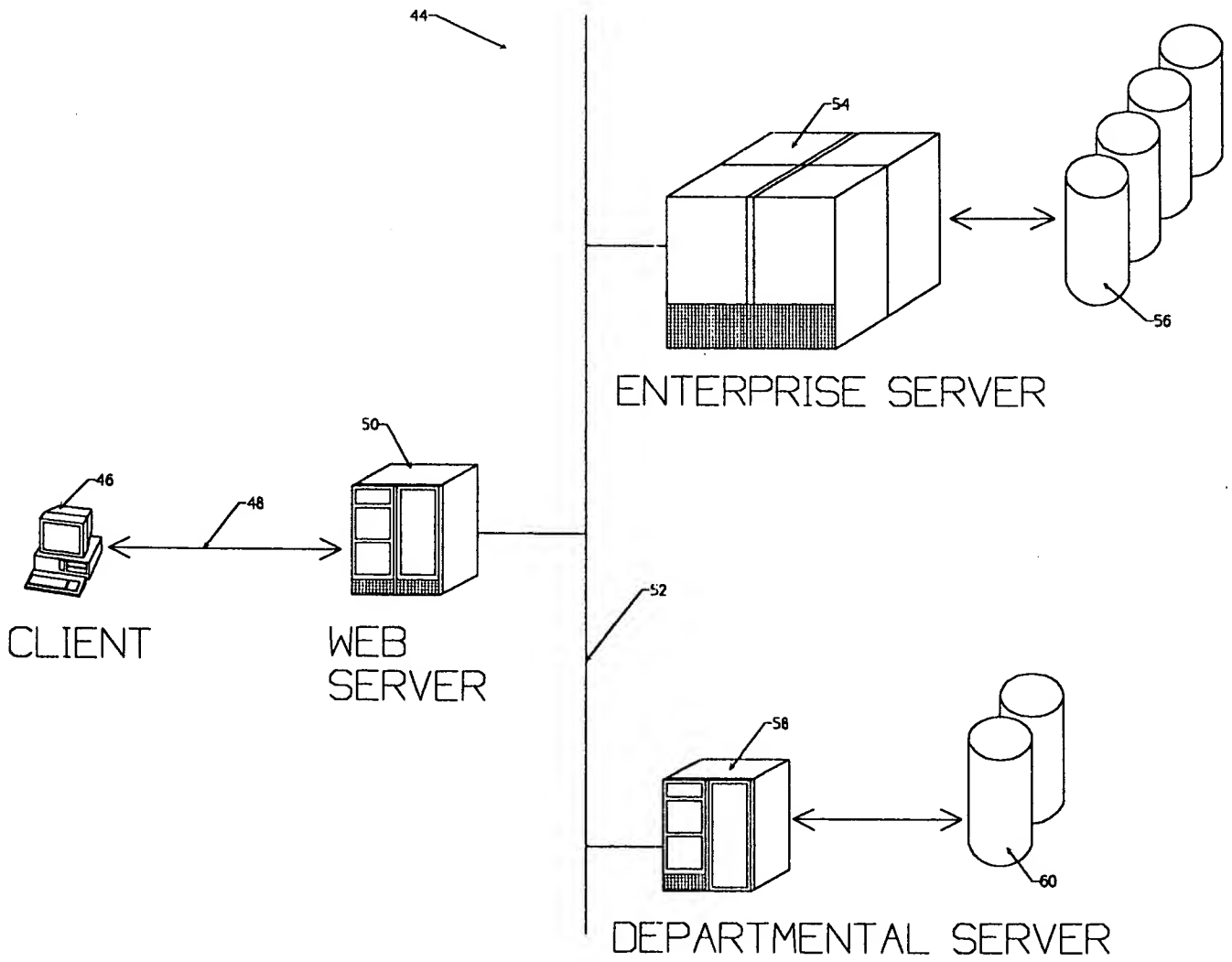
44
54
56
ENTERPRISE SERVER
50
46
48
CLIENT
WEB
SERVER
52
58
60
DEPARTMENTAL SERVER

FIG. 3

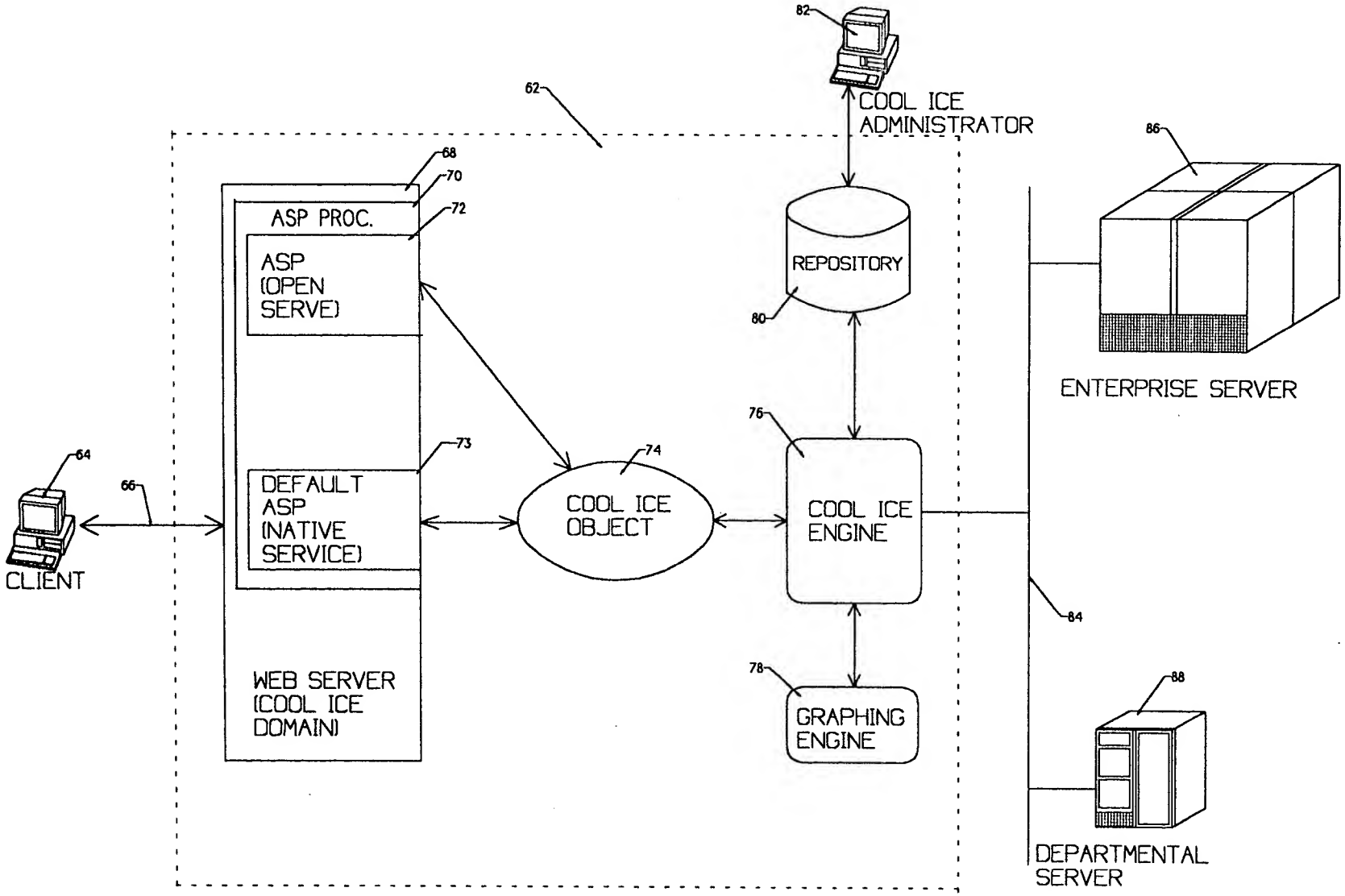

FIG. 4

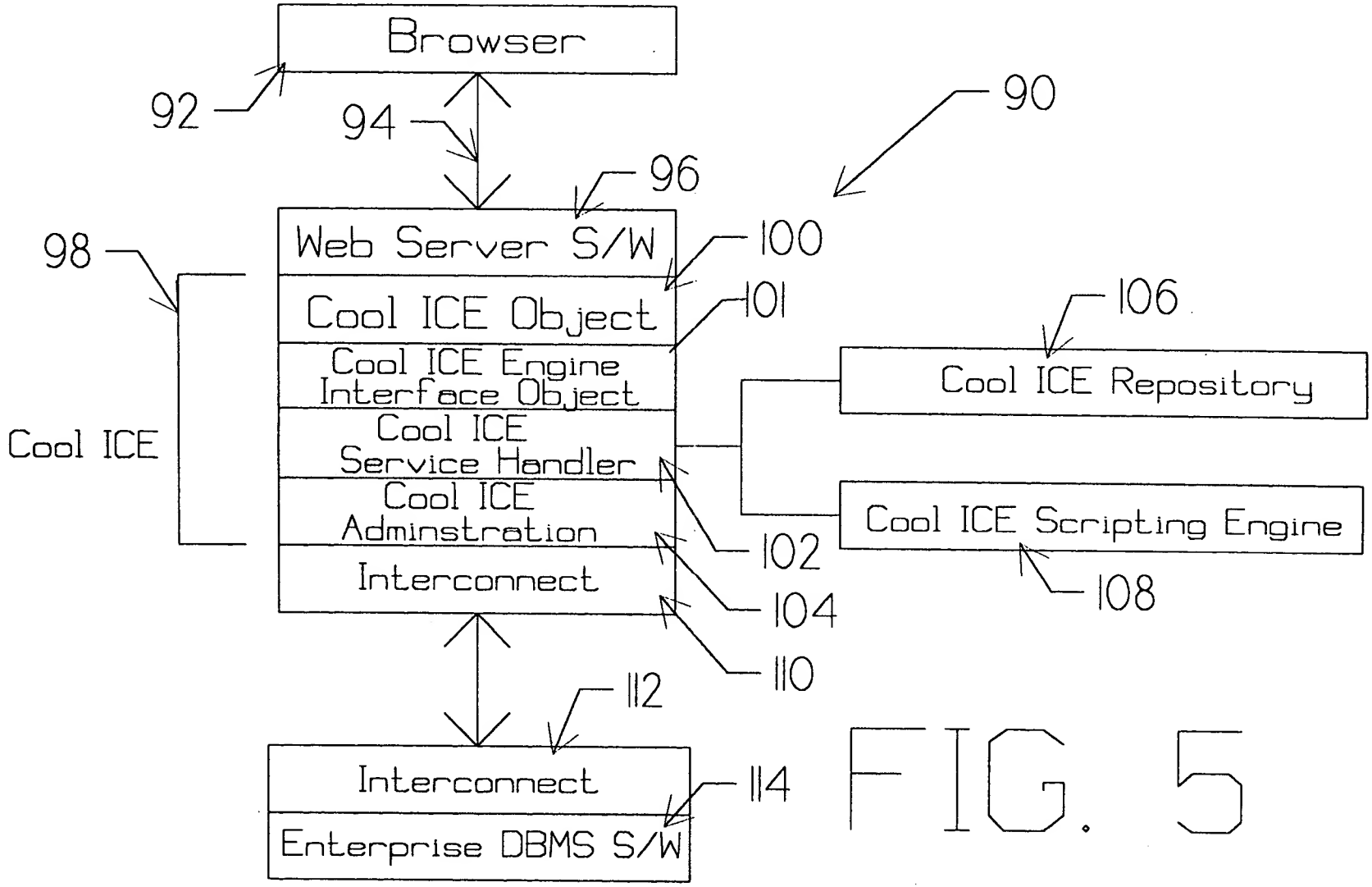
Browser
92
94
90
96
Web Server S/W
100
98
Cool ICE Object
101
106
Cool ICE Engine Interface Object
Cool ICE Repository
Cool ICE
Cool ICE Service Handler
Cool ICE Adminstration
Cool ICE Scripting Engine
102
Interconnect
104
108
110
112
Interconnect
114
Enterprise DBMS S/W

FIG. 5

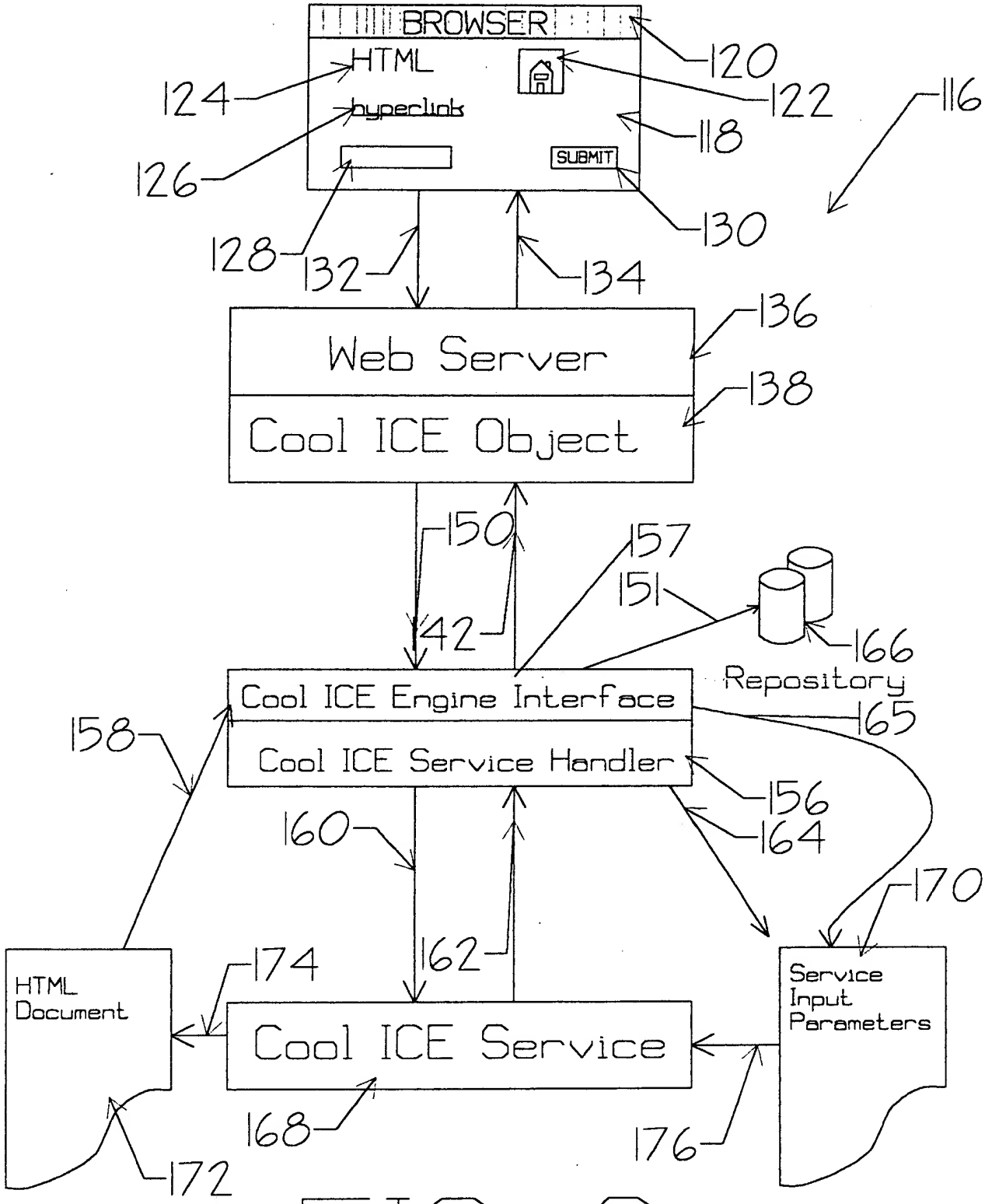
BROWSER
HTML
hyperlink
SUBMIT
120
122
124
126
116
118
128
130
132
134
Web Server
136
Cool ICE Object
138
150
157
151
142
166
Repository
Cool ICE Engine Interface
165
158
Cool ICE Service Handler
156
160
164
170
162
HTML Document
174
Service Input Parameters
Cool ICE Service
168
176
172

FIG. 6

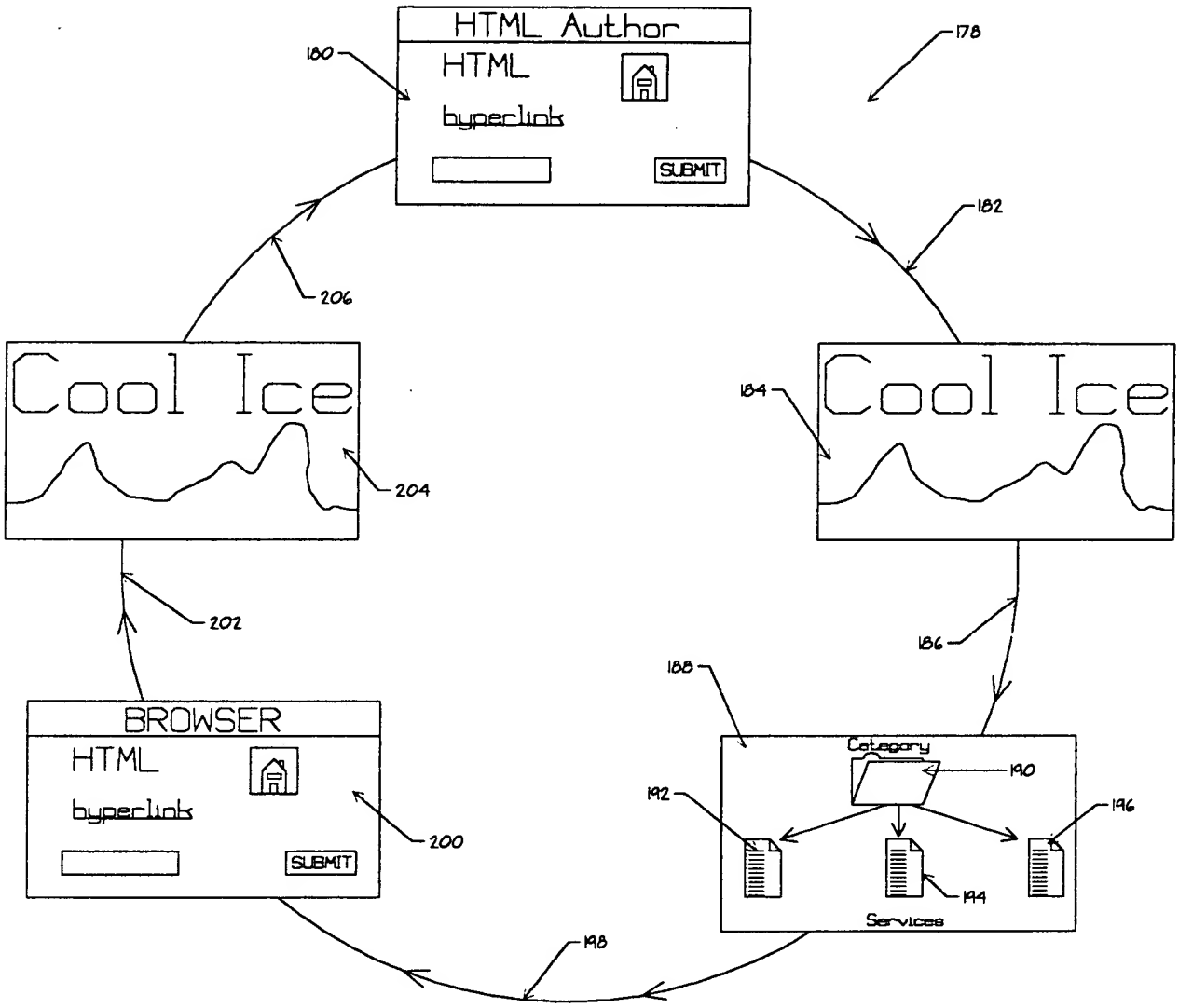
HTML Author
HTML
hyperlink
SUBMIT
178
180
182
206
Cool Ice
204
Cool Ice
184
202
186
188
BROWSER
HTML
hyperlink
SUBMIT
200
Category
190
192
194
196
Services
198

FIG. 7

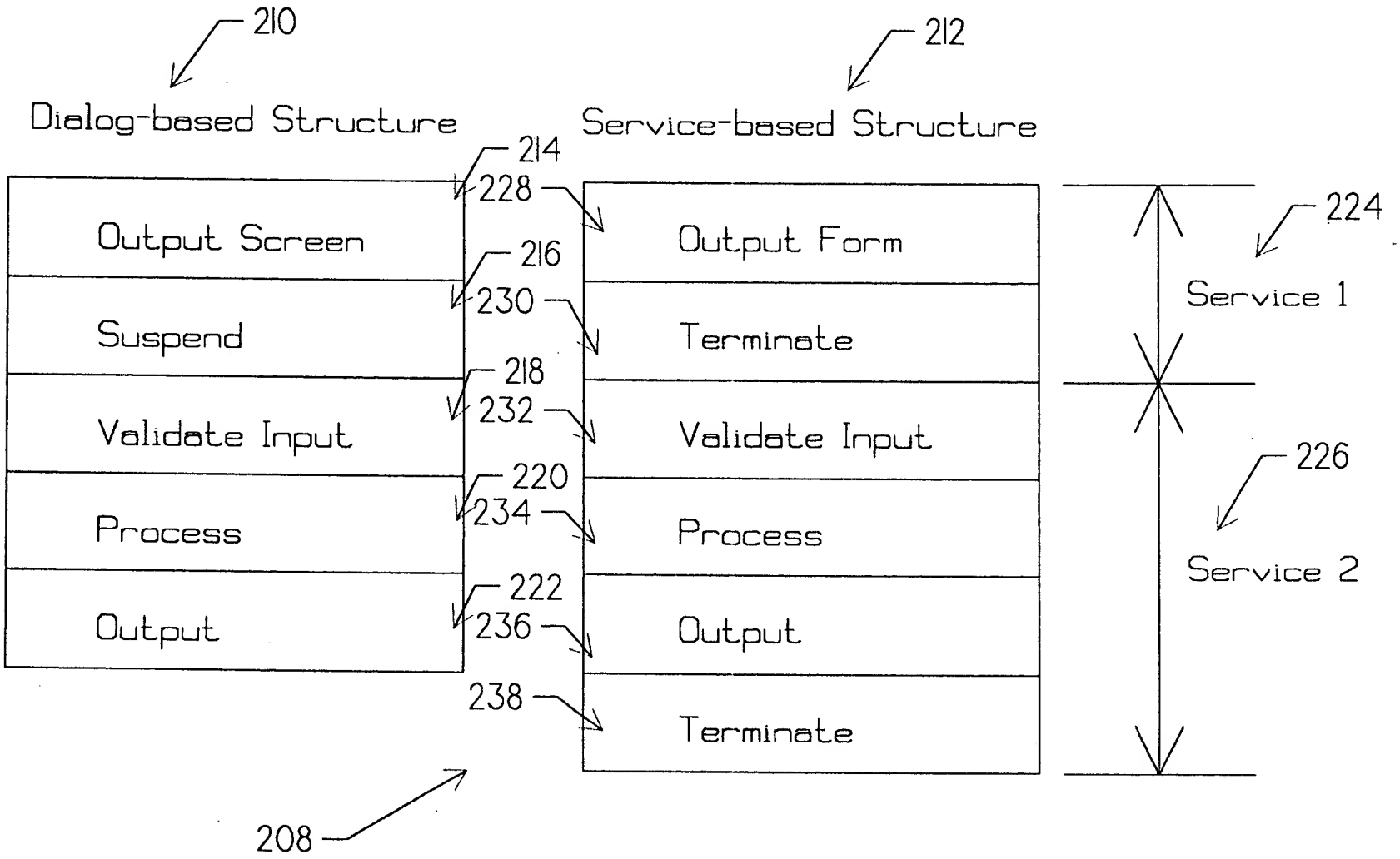

FIG. 8

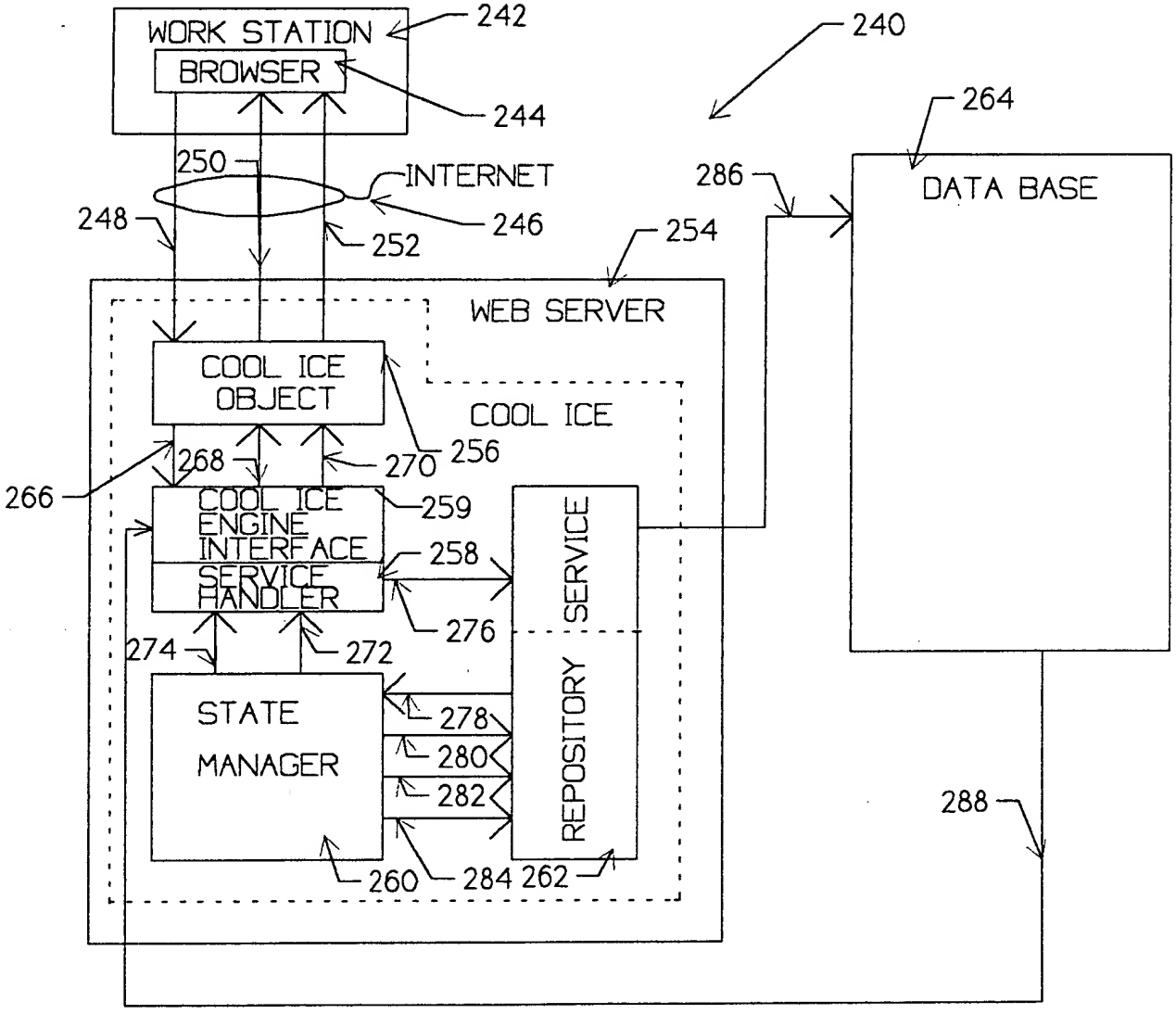
WORK STATION
242
BROWSER
244
240
264
250
INTERNET
286
DATA BASE
248
246
252
254
WEB SERVER
COOL ICE OBJECT
COOL ICE
256
268
270
266
COOL ICE ENGINE INTERFACE
259
SERVICE HANDLER
258
276
SERVICE
REPOSITORY
274
272
STATE MANAGER
278
280
282
288
260
284
262

FIG. 9

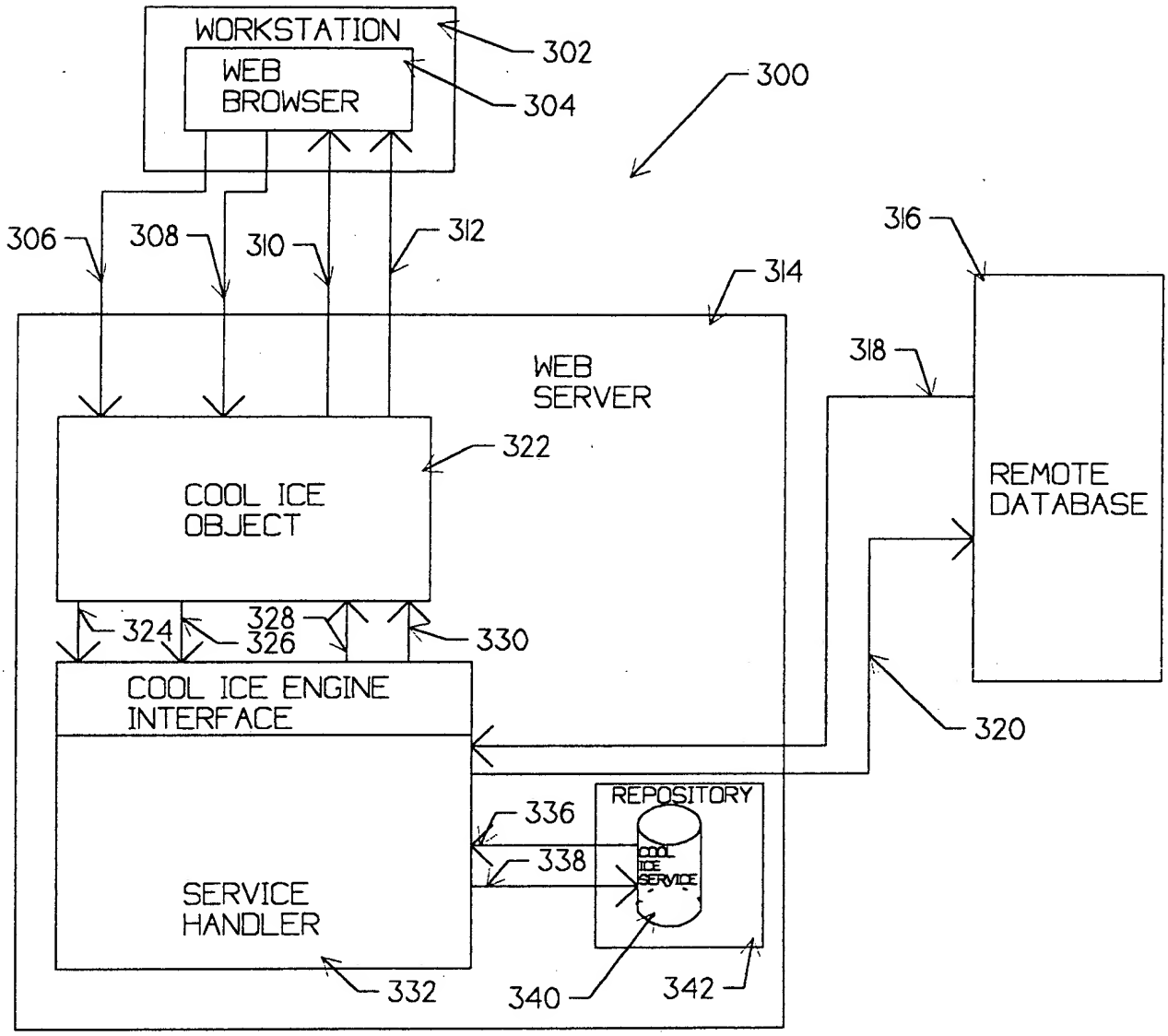
WORKSTATION
WEB BROWSER
302
304
300
306
308
310
312
316
314
WEB SERVER
318
322
COOL ICE OBJECT
REMOTE DATABASE
324
328
326
330
COOL ICE ENGINE INTERFACE
320
REPOSITORY
336
338
COOL ICE SERVICE
SERVICE HANDLER
332
340
342

FIG. 10

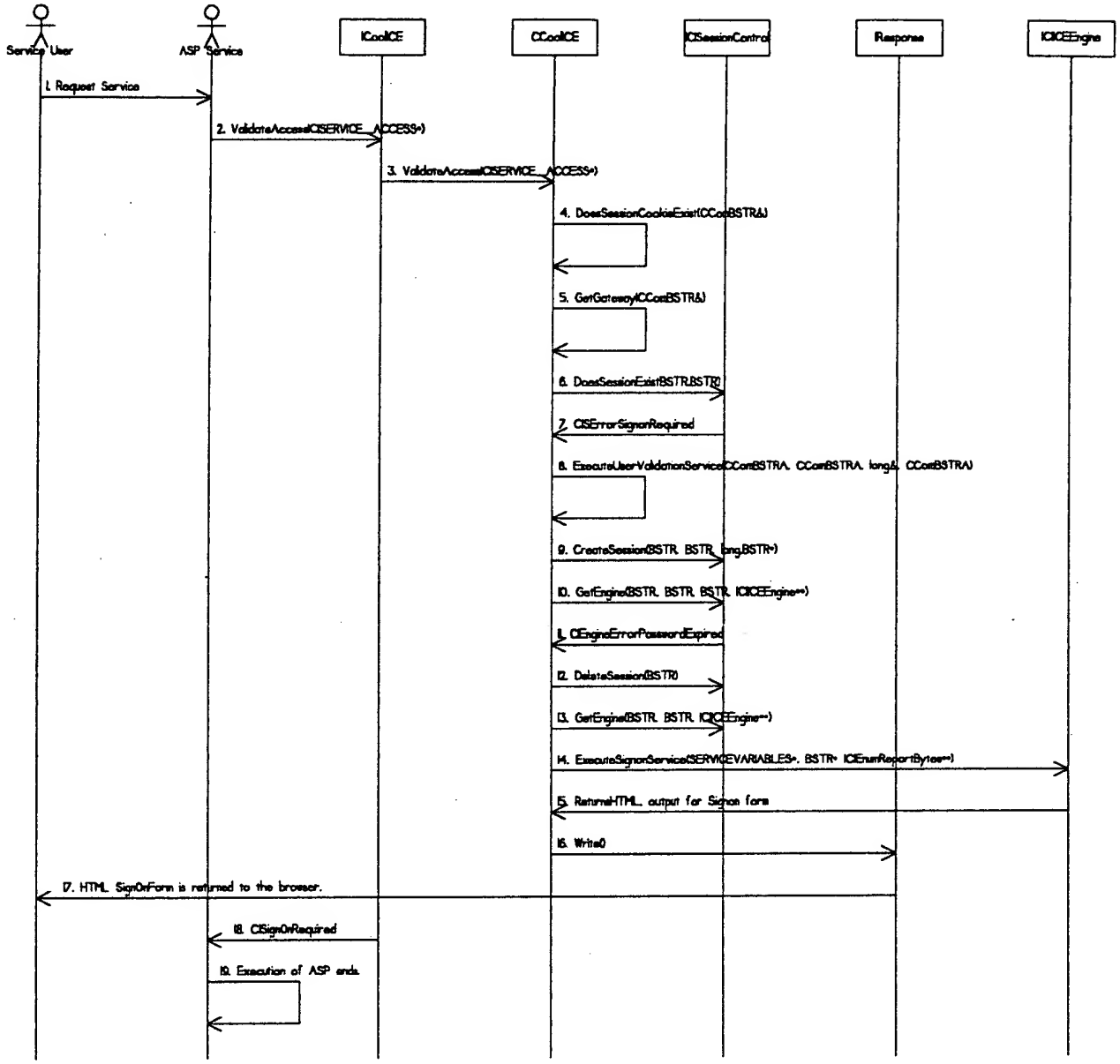

FIG. 11

MESSAGE #	DESCRIPTION
1.	The Service User will make a request for an ASP page from a browser.
2.	The ASP is executed which will create a CoolICE object and call the ValidateAccess0method. This method is intended to validate that the Service User does have access to the ASP being executed.
3.	
4.	Call the helper method DoesSessionCookieExist to determine if the CSESSIONID cookie exists in the Request.Cookies collection. In this sequence diagram assume S_FALSE is returned.
5.	GetGateway is called to extract the Gateway Name from the ASP ServerVariables Collection of the ASP Request object. The PATH_INFO variable will return the part of the URL after the server name but before any query string. The gateway name would be the first directory in the PATH_INFO. For example: URL Request: http://MyServer/CoolICE/doc.asp PATH_INFO:/CoolICE/doc.asp Gateway: CoolICE
6.	DoesSessionExist0 is called to determine if a the HTML SignOn form needs to be processed. The bstrSessionID parameter is set to an empty BSTR. The bstrGatewayName parameter is the value returned by GetGateway0.
7.	DoesSessionExist0 has determined that a COSession object does not exist and a signon is required. The CSICErrorSignonRequired HRESULT from DoesSessionExists0 is described in sequence diagrams SC02 SC04, SC05, and SC07.
8.	Call ExecuteUserValidationService to process the SignOn form input fields. In this sequence diagram assume S_OK is returned which indicates that a UserID, Department, and Password are returned. Optionally, a New Password may also be returned.
9.	CreateSession0 is called to create a COSession object. The parameters are set as follows: -bstrGatewayName is the value returned by GetGateway0 -bstrUserID is the value of the bstrUserID parameter from the call to ExecuteUserValidationService0 -bstrPassword is the value of the bstrPassWd parameter from the call to ExecuteUserValidationService0 -nDepartment is the value of the nDept parameter from the call to ExecuteUserValidationService0 -pbstrSessionID is the address of a local variable.
10.	In order to validate that the Service User has access to the ASP, a CoolICE engine is required. In addition, if the Service User does have access, then the CoolICE engine will be needed to allow the ASP to execute additional CoolICE services. Therefore, ICISessionControl::GetEngine0 is called to access an instance of a CoolICE engine that is managed by a Connection Pool. The bstrSessionID parameter is a unique identifier for the Service User. This identifier is returned by the ICISessionControl::CreateSession0 method. The bstrGatewayName parameter is the value returned by GetGateway0. The bstrNewPassword parameter is the value of he bstrNewPassword parameter returned by ExecuteUserValidationService0. In most cases, this parameter will be an empty string, except when the current password has expired, and a new password was specified.
11.	The call to GetEngine has returned the HRESULT value CEngineErrorPasswordExpired which indicates that the specified password has expired. A special SignOn form must now be displayed which allows the user to change their password.
12.	Since an Error was returned by GetEngine, it is necessary to delete the CSession object created in step 9 above. Pass the bstrSessionID value that was returned by CreateSession.
13.	It is necessary to get a Cool ICE Engine so that the SignOn service can be executed. In this case, a blank SessionID should be specified so that the default user-id/department/password will be used to sign on to the Cool ICE engine. The bstrSessionID parameter should be a blank string so that the default user-id/department/password will be used to sign on to the Cool ICE engine. The bstrGatewayName parameter is the value returned by GetGateway0. The bstrNewPassword parameter should be a blank string since we know that a new password has not been specified.
14.	The ExecuteSignOnService0 method is called to cause the SignOn service to be executed. This service will return a HTML form which has the user-id and department prefilled. A field for the old password and new password must be filled in by the user. The input parameters 'user' and 'dept' must be passed to the SignOn service when a password has expired. The parameters are passed in the BrowserInput section of the SERVICEVARIABLES structure. The values of these parameters are the bstrUserID and nDept parameters returned by ExecuteUserValidationService0.
15.	The SignOn service returns a HTML form that prompts the user for the old password and new password. The user-id and department number fields are prefilled and read-only.
16.	The HTML output for the SignOn service is passed to the Write0 method of the ASP Response object.
17.	The ASP Response object will send the HTML output for the SignOn form to the browser.
18.	ValidateAccess0 returns a CISignOnRequired status back to the ASP. This will cause the ASP to terminate.
19.	Execution of the ASP is terminated due to the CISignOnRequired status being returned from ValidateAccess0.

FIG. 12

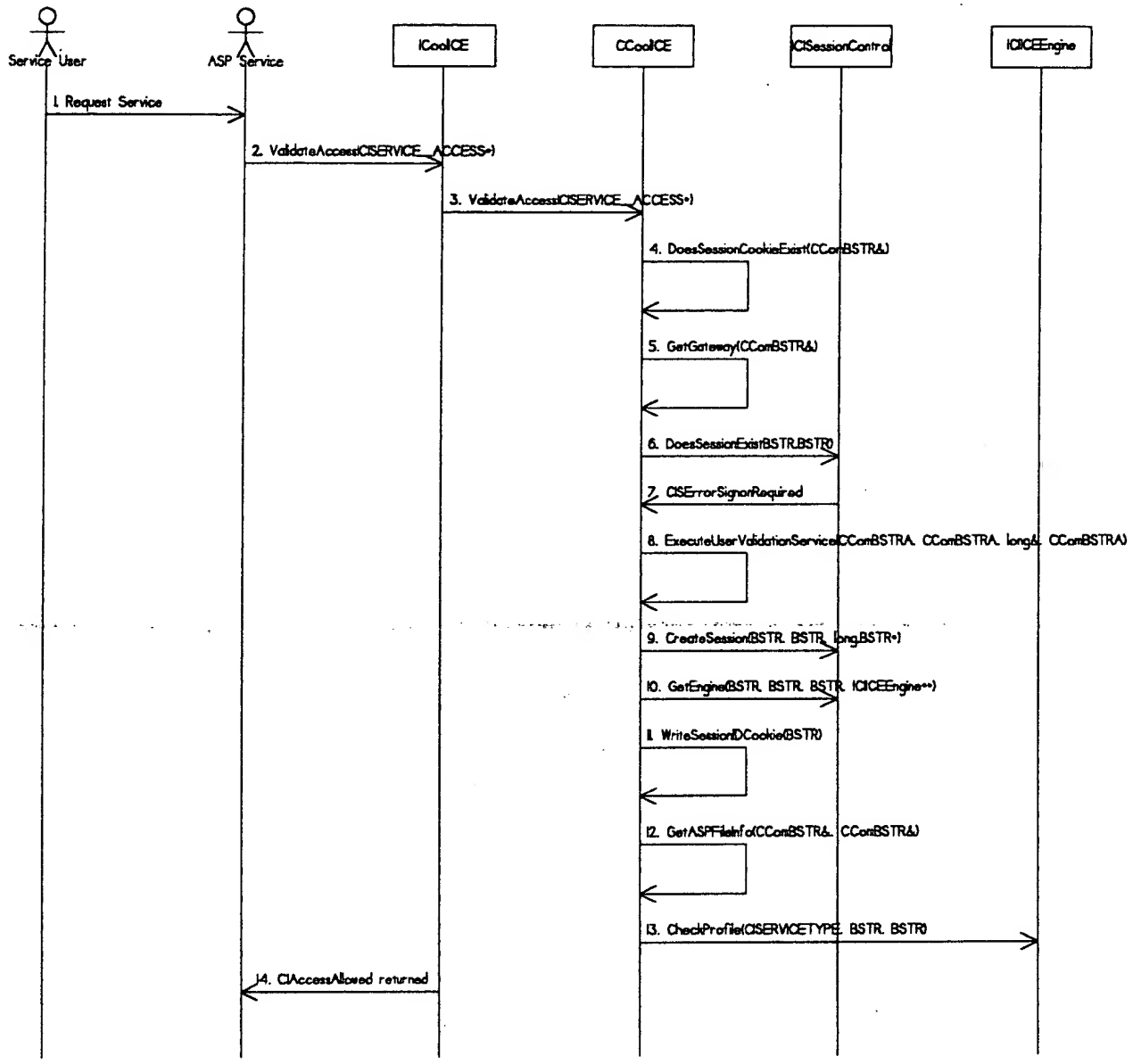

FIG. 13

MESSAGE #	DESCRIPTION
1.	The Service User will make a request for an ASP page from a browser.
2.	The ASP is executed which will create a CoolICE object and call the ValidateAccess() method. This method is intended to validate that the Service User does have access to the ASP being executed.
3.	
4.	Call the helper method DoesSessionCookieExist to determine if the CISESSIONID cookie exists in the Request.Cookies collection. In this sequence diagram assume S_FALSE is returned.
5.	GetGateway is called to extract the Gateway Name from the ASP ServerVariables Collection of the ASP Request object. The PATH_INFO variable will return the part of the URL after the server name but before any query string. The gateway name would be the first directory in the PATH_INFO. For example: URL Request: http://MyServer/CoolICE/abc.asp PATH_INFO:/CoolICE/abc.asp Gateway: CoolICE
6.	DoesSessionExists() is called to determine if a the HTML SignOn form needs to be processed. The bstrSessionID parameter is set to an empty BSTR. The bstrGatewayName parameter is the value returned by GetGateway():
7.	DoesSessionExist() has determined that a CCISession object does not exist and a signon is required. The CISCErrorSignonRequired HRESULT from DoesSessionExists() is described in sequence diagrams SC02 SC04, SC05, and SC07.
8.	Call ExecuteUserValidationService to process the SignOn form input fields. In this sequence diagram assume S_OK is returned which indicates that a UserID, Department, and Password are returned. Optionally, a New Password is returned.
9.	CreateSession() is called to create a CCISession object. The parameters are set as follows: -bstrGatewayName is the value returned by GetGateway() -bstrUserID is the value of the bstrUserID parameter from the call to ExecuteUserValidationService() -bstrPassword is the value of the bstrPassWd parameter from the call to ExecuteUserValidationService() -nDepartment is the value of the nDept parameter from the call to ExecuteUserValidationService() -pbstrSessionID is the address of a local variable.
10.	In order to validate that the Service User has access to the ASP, a CoolICE engine is required. In addition, if the Service User does have access, then the CoolICE engine will be needed to allow the ASP to execute additional CoolICE services. Therefore, ICISessionControl::GetEngine() is called to access an instance of a CoolICE engine that is managed by a Connection Pool. The bstrSessionID parameter is a unique identifier for the Service User. This identifier is returned by the ICISessionControl::CreateSession() method. The bstrGatewayName parameter is the value returned by GetGateway(). The bstrNewPassword parameter is the value of he bstrNewPassword parameter returned by ExecuteUserValidationService(). In most cases, this parameter will be an empty string, except when the current password has expired, and a new password was specified.
11.	Call WriteSessionIDCookie to write the SessionID value returned by ICISessionControl::CreateSession out to the browser as the CISESSIONID cookie.
12.	The helper method GetASPFileInfo is called to retrieve the virtual directory alias name and the file name of the ASP.
13.	The ICICEEngine::CheckProfile() method is called to verify that the user, as known to the Cool ICE engine, does have access to the ASP.
14.	The ValidateAccess() method will return CIAccessAllowed status indicating that the Service User does have access, therefore, the execution of the ASP can continue.

FIG. 14